TNO Institute for Perception

TNO Defence Research

Ka TD
P.(
3769 ZG Soesterberg
The Netherlands

Fax +31 3463 5 39 77
Telephone +31 3463 5 62 11

1

TNO-report **IZF 1992 B-5**

A.M. Schaafstal
J.M.C. Schraagen

A METHOD FOR COGNITIVE TASK ANALYSIS

Number of pages: 38

Netherlands organization for applied scientific research

TNO Defence Research consists of the TNO Physics and Electronics Laboratory, the TNO Prins Maurits Laboratory and the

CONTENTS

Report No.:	IZF 1992 B-5
Title:	A method for cognitive task analysis
Authors:	Dr. A.M. Schaafstal and Drs. J.M.C. Schraagen
Institute:	TNO Institute for Perception Group: Skilled Behaviour
Date:	July 1992
DO Assignment No.:	B 92-34
No. in Program of Work:	788.3

SUMMARY

A method for cognitive task analysis is described based on the notion of "generic tasks". The method distinguishes three layers of analysis. At the first layer, the task structure, top-level goals of a certain task are identified that have to be fulfilled during task-execution. This task structure may also be viewed as the global strategy to carry out the task. At the second layer of analysis, the local strategies (procedures) are identified by means of which values are obtained for goals in the task structure. The third layer of analysis consists of a description of the underlying domain knowledge. After a general discussion of the potentialities of the task model in predicting and diagnosing human cognitive behaviour, implications of the model for applied areas such as the development of knowledge-based systems and training, are discussed.

Rap.nr. IZF 1992 B-5

Instituut voor Zintuigfysiologie TNO
Soesterberg

Een methode voor cognitieve taakanalyse

A.M. Schaafstal en J.M.C. Schraagen

SAMENVATTING

Een methode voor cognitieve taakanalyse wordt beschreven, gebaseerd op het begrip "generieke taken". In de hier beschreven methode wordt een onderscheid gemaakt tussen drie lagen van analyse. De eerste laag, de taakstructuur, bestaat uit algemene doelen die moeten worden vervuld tijdens de taakuitvoering. Deze taakstructuur kan ook worden opgevat als een globale strategie om de taak uit te voeren. De tweede laag bestaat uit de relevante lokale strategieën die worden aangewend om de doelen zoals beschreven in de taakstructuur van een waarde te voorzien. De derde laag van analyse bestaat uit een beschrijving van de onderliggende domeinkennis. Na een algemene discussie over de mogelijkheden van het taakmodel ten aanzien van het diagnostiseren en voorspellen van menselijk gedrag worden de implicaties van het model voor toepassingsgebieden, zoals de ontwikkeling van kennissystemen en vraagstukken op het gebied van training besproken.

1 INTRODUCTION

Task analysis is a frequently used method for determining the subtasks that together constitute a particular task. Task analysis is often used as a step in the development of selection and training systems, to develop knowledge-based systems and decision-support systems, and to optimise human-machine interfaces. Traditional methods for task analysis mostly yield a description of observable behaviour. However, with the increasing automation and greater complexity of computer systems the task of the humans working with those systems becomes more "cognitive". In other words: task-relevant behaviour tends to become less observable. Operators working with Command and Control systems, or with systems in process industry, mainly process information. This information may be present in the form of knowledge that the operator has, or may be presented on computer screens or may come from colleagues. The traditional methods for task analysis are insufficient for describing these new types of tasks.

This report describes a method for cognitive task analysis based on the notion of "generic tasks". As will be described in more detail below, a key problem in cognitive task analysis is the identification of the various types of knowledge and strategies used in performing a particular task. Although this problem has not been solved yet, a promising approach is the use of the concept of generic tasks such as planning, diagnosis, and design. A generic task determines to a certain extent the knowledge and strategies used in performing that task. Therefore, classifying a particular task as an example of a more generic task helps the analyst (the person performing the task analysis) in producing a task model, that is a description of a particular task in terms of strategies and knowledge for carrying out the task. System designers and cognitive psychologists may then use the task model for developing knowledge-based systems or user interfaces.

This report will proceed as follows. Chapter 2 gives definitions of some of the terminology used throughout this report. Chapter 3 describes and evaluates a number of previous approaches for cognitive task analysis. Chapter 4 introduces a framework in which three types of knowledge are distinguished that are important for carrying out complex tasks. Chapter 5 describes two generic tasks in detail. Chapter 6 shows how to develop a task model based on the three types of knowledge described in chapter 2. Chapter 7 indicates where errors in performance are likely to be observed. Chapter 8 and 9 are geared towards applications of this method. Chapter 7 describes the application of task models to the development of knowledge-based systems. Chapter 8 discusses training issues from this viewpoint. In appendix 1, practical guidelines will be given for carrying out protocol analyses.

2 TERMINOLOGY

Below, working definitions of several terms used in this report will be given.

Strategy: specification of a set of goals to be accomplished when performing a task; the order in which the goals need to be accomplished is not always a fixed one, but may depend on specific task conditions.

Global strategy: specification of a set of task-specific goals; also: knowledge about how to decompose a particular type of task into relevant goals or subtasks, often learned through years of experience with carrying out that task. The relevant subtasks are also called the 'task structure'.

Local strategy: specification of a set of procedures for accomplishing goals, or of a set of subgoals for accomplishing goals.

Procedure: a pre-specified way of carrying out a specific task. Procedures, in contrast to strategies, are supposed to be carried out in a fixed order.

Domain knowledge: the knowledge selected by a local strategy for accomplishing a particular goal in a particular task; domain knowledge includes both factual or declarative knowledge and procedural knowledge. Procedural knowledge is not necessarily equivalent to knowledge of procedures. On the one hand, procedures may be stored declaratively, particularly in the early stages of task performance. On the other hand, procedural knowledge also includes heuristics, which are highly task-specific short-cuts for selecting relevant declarative knowledge (e.g., given such and such symptoms, this particular cause is most likely). Hence, heuristics are not procedures, for they select declarative knowledge and do not specify ways of carrying out tasks (note that the term 'heuristic' is sometimes used for denoting "an often successful way of carrying out a task"; this usage of the term heuristic corresponds to our use of global or local strategy).

Weak method: a strategy that is applicable across specific domains and across specific tasks, hence requires little domain knowledge for successful application (e.g.,
means-ends analysis, reasoning by analogy, left-most depth-first search).

Strong method: a strategy that is applicable only in specific domains and specific tasks, hence requires a lot of domain knowledge for successful application (e.g., heuristics are strong methods, because they are only applicable in specific tasks).

Task model: description of a particular task in terms of the strategies and domain knowledge required for carrying out the task.

Generic task: a class of tasks that are carried out similarly across application domains, e.g., in the diagnosis of circuits, cars, power plants, or diseases, signifi-

cant elements are in common; specifically: the same strategies and the same type of domain knowledge. This communality across domains makes diagnosis a generic task.

3 PREVIOUS APPROACHES TO COGNITIVE TASK ANALYSIS

At the outset it should be noted that we are here concerned with tasks involving large cognitive components, for instance Command and Control planning tasks. These types of tasks cannot be analyzed adequately by methods of task analysis that use a vocabulary geared toward physical activities. We will therefore restrict ourselves to methods for cognitive task analysis. For other methods for task analysis the reader is referred to Drury, Paramore, Van Cott, Grey, and Corlett (1987). Below, we will discuss two methods for cognitive task analysis: Hierarchical Task Analysis (HTA), and GOMS Task Analysis.

3.1 Hierarchical Task Analysis

Hierarchical Task Analysis (e.g., Shepherd, 1985) is a method for systematically redescribing the overall task goal into a set of subordinate goals, together with a plan which states how these subordinate goals must be carried out in order to achieve the overall goal. For instance, the overall goal of operating an overhead projector may be decomposed into the subordinate goals of:

1 Ensure standby equipment available
2 Set up projector
3 Show slides according to lecture schedule
4 Switch off projector
etc.

The subordinate goal of setting up the projector may in turn be decomposed into:

1 Ensure projector is plugged in
2 Switch on projector to ensure it is working
3 Establish correct image

A plan may specify that the three subordinate goals should be carried out in the order 1-2-3-EXIT.

HTA may be seen as a description of what needs to be achieved in the task, the state which the environment should be in when the task is completed, that is the task demands or task goals (Bainbridge, 1989). No description of the cognitive processes used by the person doing the task is provided. For instance, the subordinate goal of "Show slides according to lecture schedule" should make reference to the knowledge structure that constitutes the lecture schedule. A

lecture schedule is a knowledge structure that enables the lecturer not merely to show the slides in a fixed order, but also to estimate the remaining time, and to retrieve from long-term memory the contents of the lecture.

HTA provides no guidance on what to expect when analyzing a task. It is assumed that the analyst starts from scratch each time a task is analyzed, except that the analyst gets more experienced in HTA itself. This assumption may be too weak, given what is known from other research areas. For instance, an important assumption in expert systems development and modelling of expertise is that certain types of tasks share certain goals and methods for attaining those goals (Steels, 1990). Therefore, although HTA is a perfectly valid way of describing a task's goal structure, more direction can be given to the analyst on what goal structure to expect for a given type of task. This reduces the amount of work involved considerably.

3.2 GOMS Task Analysis

The acronym GOMS stands for Goals, Operators, Methods, and Selection Rules. The GOMS method is based on an information-processing model called the "Model Human Processor" which can be described by 1) a set of memories and processors together with 2) a set of principles, called the "principles of operation". The Model Human Processor can be divided into three interacting subsystems: 1) the perceptual system, 2) the motor system, and 3) the cognitive system, each with its own memories and processors. The perceptual system consists of sensors and associated buffer memories, the most important buffer memories being a Visual Image Store and an Auditory Image Store to hold the output of the sensory system while it is being symbolically coded. The cognitive system receives symbolically coded information from the sensory image stores in its Working Memory and uses previously stored information in Long-Term Memory to make decisions about how to respond. The motor system carries out the response. The basic quantum for cognitive processing is the recognize-act cycle, which may also be described as production rules. In this description, the production rules reside in Long-Term Memory. On each cycle, the recognition conditions of the rules are compared with the contents of Working Memory. The rule with the best match fires and causes its associated action to occur, thereby altering the contents of Working Memory. One of the interesting things of the GOMS model is the association that has been made between the memories and processors of the various subsystems and associated parameters which are used for understanding and predicting human performance, such as the processor cycle time, the memory capacity, memory decay rate, and the memory code type. For more information about the Model Human Processor, the reader is referred to Card, Moran, and Newell (1983).

A GOMS model, as proposed by Card, Moran, and Newell (1983), is a description of the knowledge a user must have in order to carry out tasks on a device or

system. Briefly, a GOMS model consists of descriptions of the Methods needed to accomplish specified Goals. The Methods are a series of steps consisting of Operators that the user performs. A Method may call for sub-Goals to be accomplished, so the Methods have a hierarchical structure. If there is more than one Method to accomplish a Goal, then the GOMS model includes Selection Rules that choose the appropriate Method depending on the context.

Below, an example Method will be given for a file manipulation task in Macintosh Finder (from Kieras, 1991):

Method for accomplishing goal of deleting a file
Step 1 Accomplish goal of dragging file to trash
Step 2 Return with goal accomplished

In addition to the specific deleting method, there is a general submethod corresponding to the drag operation: this is the basic method used in most of the Macintosh Finder file manipulations. It is called like a subroutine by the above methods.

Method for accomplishing goal of dragging item to destination
Step 1 Locate icon for item on screen
Step 2 Move cursor to item icon location
Step 3 Hold mouse button down
Step 4 Locate destination icon on screen
Step 5 Move cursor to destination icon
Step 6 Verify that destination icon is reverse-video
Step 7 Release mouse button
Step 8 Return with goal accomplished

Method descriptions such as described in the example above can be very useful for comparing systems. For example, Kieras (1991) compared the number and length of methods of the Macintosh Finder and PC-DOS. He concluded that to accomplish the same user goals in PC-DOS required four times as many steps as in the Macintosh Finder. The Method descriptions can be used to derive predictions of learning and execution time, since the user must learn these step-by-step methods in order to learn how to perform these tasks. According to research results (Bovair, Kieras, & Polson, 1990), the learning time is linear with the number of steps.

One of the advantages of the GOMS approach to task analysis is that the user's procedural knowledge is specified to such an extent that computer simulation models can be developed that can actually execute the same tasks as the user. In this way, proposed interface designs can be evaluated with simulation techniques rather than actual human user testing. Aspects of the design which are not directly related to the procedures entailed by the design, such as screen layout or system functionality, would involve additional user testing, and almost certainly

require another kind of modelling than GOMS (Olson & Olson, 1990). It would seem, therefore, that the GOMS approach is more suitable for modelling routine expert behaviour rather than novice or less routine expert behaviour (Olson & Olson, 1990). This is because routine expert behaviour can be described to a large extent in terms of procedural knowledge (Anderson, 1987). In contrast, nonskilled users spend considerable time engaged in problem-solving activities, rather than simply retrieving and executing plans. Other important aspects of human-computer interaction which are currently beyond the scope of GOMS are the impact of fatigue on performance, user acceptance, and an assessment of how computers fit into the larger picture of work and organizational roles (Olson & Olson, 1990).

Conceptually, GOMS task analysis is very similar to HTA. However, whereas HTA primarily gives a description of the task goals, GOMS gives a description of the procedural knowledge required for performing a task. GOMS includes the cognitive processes used by the person doing the task, HTA does not. Thus, GOMS task analysis starts where traditional task analysis stops. The amount of analytic detail involved is considerable, making the construction of GOMS models a lengthy and effortful task. The major difficulty for the analyst is to specify the user's goals, operators, methods, and selection rules. Kieras (1991) does not provide more guidance than that the analysts should use their own intuition about how users perform tasks, resulting in so-called "judgment calls" or intuitive assessments. However, Kieras (1991) argues that these judgment calls, since they are based on a careful consideration from the user's point of view, can do no more harm and should lead to better results, than the designer's assumptions that usually ignore the user's point of view.

In conclusion, GOMS is a useful method for task analysis for repetitive and stylized computer-based interactions and a restricted set of people (proficient users) performing those tasks. It has primarily been applied to analyze rather simple and routine word processing tasks such as deleting, moving, and copying text. In these kinds of tasks the approach is very powerful. Predictions about performance can be made for designs without having to build prototype systems and having to run extensive, time-consuming user tests. User tests can, of course, reveal other things such as errors, problem solving, and initial learning and representation difficulties; however, cognitive models can screen out certain classes of poor designs for these kinds of tasks. It is doubtful whether GOMS can be applied to higher cognitive processes such as planning, diagnosis, and design, since these tasks involve more types of knowledge than just procedural knowledge.

3.3 Conclusion

The discussion of two well-known methods for task analysis has revealed two limitations:

1 More types of knowledge are usually involved in higher cognitive processes than merely procedural, how-it-works, knowledge. This is particularly true in novel task situations, where even experts will use domain principles, and a range of weaker problem solving strategies, as indicated by recent cognitive research (see Holyoak, 1990, for a review). GOMS, with its heavy emphasis on procedural knowledge, will be difficult to apply to, for instance, Command and Control planning tasks.
2 Both HTA and GOMS do not provide the analyst much guidance in specifying what goals, operators and methods are involved in particular tasks. More guidance can be given if we accept the notion of generic tasks, that is, classes of tasks that share particular goal and subgoal structures and methods.

We will take up these two limitations in the next two chapters. First, chapter 4 will present a framework in which several types of knowledge, not just procedural knowledge, can fit. Second, chapter 5 will present two generic tasks that together cover a wide range of tasks frequently encountered.

4 TYPES OF KNOWLEDGE: A THEORETICAL FRAMEWORK

Many researchers make a distinction between declarative and procedural knowledge (e.g. Anderson, 1983; 1987). Declarative knowledge may be conceived as a collection of stored facts and is also called system or device knowledge in the domain of technical systems. Examples of this are knowledge about normal values of certain parameters, or knowledge about the function of the system. Procedural knowledge (knowledge about how-to-do-it) can be regarded as a collection of actions or procedures that an intelligent system can carry out. It also contains knowledge of the procedures with which one investigates a device to make diagnoses about its functioning, for example the use of the oscilloscope to test certain functions of a system. Procedural knowledge is content-specific and thus only applicable in a limited domain.

In addition to the declarative-procedural distinction, a distinction can be made between domain-specific knowledge and strategic or metacognitive knowledge. This strategic or metacognitive knowledge (knowledge about how-to-decide-what-to-do-and-when) is applicable across specific content domains, but remains geared towards one task (e.g. diagnosis). For example, in diagnosis, regardless of the domain, one would first *identify and interpret symptoms*, followed by *an investigation of possible reasons*, which will be *tested*, before one will apply a certain *repair or remedy*.

At a very general level, problem-solving strategies may be identified which have to do with very general thinking and reasoning skills, such as means-ends analysis, reasoning by analogy, or working backwards. These general problem-

solving strategies, or *weak methods* as they are sometimes called, are applicable across specific domains and across specific tasks.

With this decomposition, it is assumed that *procedural* and *device* knowledge are organized and deployed by goals, plans and decision rules that comprise *strategic* knowledge. Thus, strategic knowledge can be said to have a *control* function to enable dynamic, flexible reasoning. As described in Gott (1989), support for the concept of strategic control knowledge comes from a number of academic domains such as geometry (Greeno, 1978), text editing (Card, Moran, & Newell, 1983), computer programming (Anderson, Boyle, Farrell, & Reiser, 1984), simple device operation (Kieras & Bovair, 1984) and electronic troubleshooting (Gott & Pokorny, 1987). The question is though, whether this strategic knowledge, being general in nature, can be transferred to tasks in related fields, in this manner enabling intelligent systems to generalize across domains. An example of a recent attempt to build a system that is able to generalize across domains, and in which the interaction between domain specific and domain independent strategic knowledge is explored, is FERMI, an expert system that reasons about natural science domains (Larkin, Reif, Carbonell, & Gugliotta, 1988). In FERMI, domain specific knowledge of scientific principles and strategic problem solving knowledge are encoded in separate but related semantic hierarchies. This allows the system to apply common problem solving principles of invariance and decomposition as encoded in the strategic problem solving knowledge base to a variety of problem domains such as fluid statics, DC-circuits, and centroid location. Similarly, in the knowledge based tutoring system Guidon for a medical domain (Clancey, 1987), it turned out to be very important to separate tutoring knowledge, and knowledge about general strategies, from specific domain knowledge. Processes of acquiring strategic knowledge have been addressed in analyses by Anzai and Simon (1979) on the Tower of Hanoi and by Anderson, Farrell, and Sauers (1984) on the acquisition of knowledge needed to learn Lisp. As stated in Greeno and Simon (1988), both studies showed that important factors in acquiring strategic knowledge are the activation of a problem goal that can be achieved by a sequence of actions and the acquisition of productions in which the action of setting the goal is associated with appropriate conditions in the problem situation. The importance of strategic knowledge has been shown by Greeno (1978). He conducted an experiment on the acquisition of high school geometry knowledge. As the resulting computational model PERDIX of problem solving behaviour showed, strategic knowledge is needed for setting goals that organize problem-solving activity. One of the students in Greeno's study knew the problem-solving operators and the geometric patterns to achieve them, but was unable to solve the problem. This result can be explained by the hypothesis that the student lacked knowledge about the problem solving strategy needed in this problem. Schoenfeld (1979) showed that students who were given explicit instruction in the use of heuristic strategies in the domain of college mathematics, showed superior performance compared to students who had not received this training.

A difference exists between strategic knowledge as implemented in PERDIX or as trained in Schoenfeld's study, and strategies as used by, for example, GPS (Ernst & Newell, 1969) or Soar (Laird, Newell, & Rosenbloom, 1987). In GPS and Soar, only very general problem-solving strategies, or so-called weak methods such as means-ends analysis, have been implemented. These weak methods, as stated before, do not contain any domain-specificity. In PERDIX and in Schoenfeld's study the emphasis is put upon a generic strategy for a class of problems, such as diagnosis or geometry problems. Thus, the strategies as researched in PERDIX and by Schoenfeld are weak in the sense that they do not contain domain-specific knowledge. However, they are stronger than weak methods such as means-ends analysis, in the sense that they are task-specific.

Thus, if the above results are put together, the following framework emerges. At the top-level strategic knowledge is employed, consisting of several goals that have to be fulfilled during task execution. Examples of these goals in the diagnostic task may be "Identify symptoms", "Determine possible causes", and "Identify repair". As the reader may have noticed, these goals may also be viewed as various subtasks of the diagnostic task. This layer of knowledge is called knowledge about the *task structure*, or the *global strategy* (Wognum, 1990). The order in which those goals are fulfilled is often flexible, and depends on the specific task-situation at hand. Each goal, however, only defines *what* intermediate conclusion has to be deduced, not *how* the intermediate conclusion has to be deduced. The knowledge about how to perform a (sub)task is called the *local strategy*, and may consist of either procedures with a fixed order (how to test a certain part), or may consist of a more flexible sequence of steps: a strategy. The local strategy of a subtask in turn selects the *domain knowledge* necessary to achieve the goal of the task. If a person becomes very experienced with the task at hand, he may use heuristic shortcuts in his reasoning process to obtain values for the various goals determined at the task level. For example: if that symptom occurs, it is rather likely that it is caused by that fault. The heuristics are bound to the task at hand: a person very experienced in one domain, which may turn him into an efficient problem-solver regarding that domain, does not necessarily show that same proficiency in other domains. On the other hand, if a person has no experience at all with the task at hand, he is bound to use very general reasoning methods: weak methods.
In a cognitive task analysis all three levels of knowledge, the global strategy or task structure, the local strategies, and the domain knowledge, have to be filled in for a specific domain. These three types together constitute the *task model* of a task in a given domain.

The information-processing model underlying this framework is taken from a production system model, such as ACT* (Anderson, 1983, 1987). Important features of ACT* in this context are the following:

- *Productions form the units of knowledge.* Productions define the steps in which a problem is solved and are the units in which knowledge is acquired.

- *Hierarchical goal structure.* The ACT* production system specifies a hierarchical goal structure that organizes problem solving. The hierarchical goal structure closely reflects the hierarchical structure of the problem. Goals are also important in structuring the learning by knowledge compilation. They serve to indicate which parts of the problem solution belong together and can be compiled into a new production.
- *Initial use of weak methods.* The use of weak methods, such as analogy or means-ends analysis, plays a critical role in getting initial performance of the ground.
- *Knowledge compilation.* All knowledge in the ACT* theory starts out in declarative form and must be converted into a procedural format. This declarative knowledge can be encodings of instructions, encodings of general properties of objects, and so on.
- *Limited working memory.* Working memory is limited in the sense that it can only hold a limited amount of information at a particular point in time.

The implications of this information-processing model in relation to the described method for cognitive task analysis are put forward in chapter 7.

5 MODELS FOR GENERIC TASKS

The notion of generic tasks first arose in the context of developing an engineering methodology to build expert systems based on a task analysis. Several researchers observed that tasks fall into major classes. These tasks are called generic tasks (Chandrasekaran, 1983). In specific fields of expertise, tasks are instances of these generic tasks. Typical generic tasks are classification, interpretation, diagnosis, and construction (including planning and design). The way in which generic tasks are executed shows many similarities across application domains: In the diagnosis of circuits, cars, power plants, or diseases, significant elements are in common, specifically, the same problem-solving methods and the same type of domain models (Steels, 1990). A task is characterized in terms of the problem that needs to be solved. This characterization is based on properties of the input, the expected output, and the nature of the operation taking place to map input to output. For example, if the input consists of observed symptoms, and the output is an explanation of how the symptoms came about, then we characterize the task as diagnosis (Steels, 1990).

There are now several descriptions of generic tasks available. We will summarize what is known about two generic tasks: diagnosis and construction (including planning and design). For a description of the generic task of interpretation, the reader is referred to Steels (1990).

5.1 Diagnosis

Of all generic tasks, diagnosis has probably been studied most extensively, both from an Artificial Intelligence (AI) and a psychological point of view. The AI perspective has focused on how diagnosis should be carried out in order to accomplish particular task goals, whereas the psychological perspective has focused on how people actually carry out diagnostic reasoning. When analyzing a particular task, it is very often a good strategy to first start with a normative task description, and later note the way people deviate from this description. However, the normative task description itself is very often based on human (expert) task performance, so the distinction between the normative and descriptive approaches is not as absolute in practice as it is in theory. Still, we will first describe the diagnostic task in a normative way, before turning to psychological deviations from the norm.

At the highest level, diagnosis consists of the following three tasks (Wognum, 1990):

1 Generate hypotheses
2 Refine hypothesis
3 Pursue hypothesis

In diagnosis, the first task is to associate observed symptoms with possible hypotheses that may explain those symptoms. The second task is to refine each of these hypotheses further, if necessary. This is particularly likely when hypotheses are stated at a rather abstract level, so that various more specific hypotheses may be distinguished. The third task is to test each hypothesis until all hypotheses have been considered. A hypothesis that has been rejected is not considered any further, whereas when a hypothesis has been confirmed it needs to be refined until the lowest level in a taxonomy has been reached.

Each task is performed by local strategies that select particular domain knowledge. For instance, the task Generate hypotheses takes as input patient findings (in the case of medical diagnosis). A local strategy then selects knowledge which associates findings with diseases. The task Refine hypothesis takes a disease as input. A local strategy then selects knowledge that finds subcases for the disease in the disease taxonomy. For instance, the hypothesis heart disease is refined in this way into myocardial infarction and infection of the heart muscle. The task Pursue hypothesis tests a particular hypothesis by using knowledge that associates diseases with their causes and with characteristic findings indicative for the hypothesis.

Of course, other types of knowledge than the ones mentioned above may be used as well. For instance, geometric knowledge representing the spatial relations between components is often used in technical domains. Knowledge about test procedures, about normal or reference values, about the frequency of hypotheses, and about how to control an installation may all be used in diag-

nosis. These types of knowledge need not be confined to diagnosis alone, however. An architect designing a building may also use geometric knowledge.

Psychological evidence (e.g., Schaafstal, 1991) has shown that diagnosis in real-life tasks is more complex than the AI-models often assume. The following global strategy for diagnosis is proposed by Schaafstal (1991):
1 Identification of symptoms
2 Judgment: How serious is the problem
3 Determination of possible faults
4 Ordering of faults according to likelihood
5 Testing
6 Determination of remedies or repairs
7 Determination of the consequences of the application of repairs
8 Evaluation: Has the situation improved?

Schaafstal (1991) has shown that expert operators follow the steps in the model very much in the order as described above. Novice operators, on the other hand, do not make judgments about the seriousness of the problem, do not determine the consequences of the application of repairs, and do not evaluate their diagnostic reasoning. Also, novices jump much more quickly to repairs, without testing whether a certain repair is actually right for a certain situation.

An important extension of Schaafstal's model to traditional AI-models is the inclusion of repair as an integral part of diagnosis. Previous approaches to diagnosis have often limited diagnosis to fault localization. However, in real-life tasks determination of repairs is an important part of diagnosis.

5.2 Construction

Construction includes both planning and design. Planning and design differ in that planning is more concerned with the organization of actions in time, whereas design is more concerned with the organization of actions in space.

Independent AI-research in various domains has found that construction tasks are often decomposed into the following subtasks (Brown & Chandrasekaran, 1986; Malhotra, Thomas, Carroll, & Miller, 1980; Marcus, Stout, & McDermott, 1987; Mittal, Dym, & Morjaria, 1986):
1 Test specifications for incompleteness or inconsistency
2 Generate or extend a partial solution
3 Test the adequacy of the solution by matching it with constraints
4 Refine the solution by resolving violated constraints

Hence, the input to construction is a set of specifications. The output is an object that conforms to these specifications. By means of the subtasks mentioned above, the input is mapped to the output. Some of the pragmatic problems (Steels,

1990) associated with construction tasks are the incompleteness of the specifications, the large number of partial solutions possible, and the limited memory available for storing structure. These pragmatic problems determine to a large extent the strategies and types of domain knowledge used by problem solvers. For instance, the incompleteness of the specifications forces the problem solver to test the specifications by validating the data, broadening or restricting the context, classifying the data, or deducing additional features based on class membership. The large number of partial solutions possible implies a structuring of solutions in terms of typical features and not in terms of necessary and sufficient conditions. The limited memory available forces the problem solver to progressively deepen the solution.

Empirical studies (e.g., Goel & Pirolli, 1989; Hamel, 1990; Schraagen, 1990) have confirmed some of the pragmatic problems hypothesized by Steels (1990). Hamel (1990) has studied a prototypical construction task: architectural design. Based on think aloud protocols of practising architects, the following model of the cognitive activities in the architectural design process was put forward:

1 Gathering information
2 Separating the problem into parts (decomposition)
3 Solving partial problems
4 Synthesizing partial solutions
5 Moulding the result into a design based on aesthetic and professional values

The model proved to be a good description of the architectural design process in terms of the cognitive activities it comprises, and the sequence of the activities in the design process. The model differs from the AI-models discussed above in that it puts more emphasis on the synthesis of partial solutions and on aesthetic and professional values. Otherwise, the models are very similar. Hamel (1990) argues that designing may be regarded as a skill, which can be distinguished from other skills in that it entails an intricate organization of activities, an extensive decomposition of the problem, and a synthesis of partial solutions into a solution for the problem as a whole.

Experts in particular spend a great deal of time testing design specifications before generating a partial solution. Moreover, the partial solutions often come in the form of "skeletal plans" (Friedland & Iwasaki, 1985), or hierarchically organized prototypes. However, these prototypes also appear to play a role in diagnostic and planning tasks, so that they do not constitute a distinguishing characteristic of construction tasks.

Essens and McCann (1991) studied a typical planning task: deciding in what order grocery items should be put into a shopping basket. They described the following efficient procedure for executing their planning task:

1 Preplan
2 Orient
3 Adopt strategy

4 Create partial plan
5 Evaluate partial plan
6 Postplan (consolidate plan parts and review entire plan)

The steps 2 to 5 in the procedure are executed iteratively until the plan is sufficiently detailed in that it matches all constraints. In their model, this results in three levels of plan creation: a very rough plan, a rough plan, and a detailed plan.

The Essens and McCann (1991) model is very similar to the other construction models we have discussed above. The only element that is markedly different at first sight is "Adopt strategy". This process deduces plan characteristics appropriate for each level of plan creation. It sets the stage, so to speak, for the subsequent creation of a partial plan. Hamel (1990) described a similar process in his model, which we left out above for reasons of clarity and brevity. In Hamel's model, for instance, the synthesis of partial solutions is under the control of an analysis schema that solves the partial problems. This control is similar to what Essens and McCann (1991) referred to as "Adopt strategy".

5.3 Conclusion

The discussion of two generic tasks has shown that AI-models often prove good starting points for constructing a model of the task structure or the global strategy. However, empirical research in the form of protocol analyses is often required in order to make the model more realistic psychologically. Psychological realism is required when the model is to be applied for purposes of developing knowledge-based systems or successful user interfaces. Else, the user would be confronted with systems that do not take the user's task representation into account. This could lead to errors or suboptimal task performance.

The important point about generic tasks for the purpose of cognitive task analysis, is that they provide the analyst guidance in specifying the global strategy that people use when performing certain types of tasks. This is a major advantage in comparison with existing methods for task analysis, that often leave the analyst empty-handed as far as specifying task goals is concerned (see Grant & Mayes, 1991, for a recent review).

6 DEVELOPMENT OF A TASK MODEL

Chapter 4 discussed the notion of *task model* comprising the global strategy to carry out a task, the local strategies and the domain knowledge. In chapter 5 several models for the generic tasks diagnosis and construction were discussed, since the method for cognitive task analysis proposed in this report relies heavily

on the notion of generic tasks as one of the important starting points for the development of a task model for a specific task. This raises the question how one would actually develop a task model for a given task. Not surprisingly, the development of the task model consists of the development of a submodel for each of the three types of knowledge. Therefore, they will each be discussed in turn.

6.1 Development of a model of the task structure: the global strategy

The development of a model for the task structure of a given task can take two forms: either a model for the generic task is available already, which can be used as a starting point for the development of a model for current task, or there is no such model available, and one has to start from scratch.

Adaptation of an existing model for a generic task

If one has to adapt an existing model to the current situation, one of the first things to do is to conclude for oneself where and in what way the current task differs from the generic task. Are there subtasks in the generic task that seemingly do not play a role in the task at hand, or, the other way around, is the current task richer in nature, in the sense that more subtasks may be identified. This global analysis may be carried out in a variety of ways: by means of introspection (if possible): how would I carry out this task, but a better way of doing this would be to collect some preliminary verbal protocols of subjects carrying out the target task. It results in an adapted model of the task structure geared towards the task of interest.

Once the model of the task structure has been adapted this way, one should test it before using it in some application. The test of the task structure is an empirical one: do people follow the global strategy as defined by the task structure, or do they differ in some respect? Since we are concerned with the identification of strategies, the analysis of verbal protocols taken from people during task execution seem a feasible way of measuring this. A broad outline for doing a protocol analysis is given in appendix 1. What one should especially look for in the protocols is whether the different steps are taken (or subtasks carried out) and whether they are taken in the proposed ordering. Another thing that is of importance concerns the interpretability of the protocols in terms of the proposed model of the task structure: do people perform subtasks that cannot be interpreted with the model? The outcome of this test leads, if necessary, to an adaptation of the model, which might need testing again, depending on the number and severity of the changes made.

Development of a model of the task structure from scratch

The procedure for developing a model for the task structure if no generic task model is available is not very different from the procedure of the adaptation of an existing model, apart from the fact that one lacks a firm starting point. The question is how to find a starting point for a model of the task structure. As already described above, there are several ways of doing this. One could, by means of introspection (or by taking a verbal protocol from oneself), investigate the strategy used, and use that as a starting point for a further, and more rigorous, analysis of the hypothesized task structure. However, a better way of developing a first hypothesis would be to run some pilot subjects and take verbal protocols while they are doing the task. If available, one could obviously always try to find literature about a comparable task, but often this is hard to find. Once an initial model for the task structure has been developed, the testing of it proceeds in the same manner as described above.

6.2 Development of a model for the local strategies

Once a model for the task structure has been developed by means of a protocol analysis, one should investigate which local strategies are used for each of the defined subtasks. The previously acquired protocols may become of use in this stage as well. The question posed would be "How is a certain defined subtask accomplished, by means of which procedures?" Since there are no real firm insights into generic local strategies or procedures to carry out a certain subtask, except for diagnosis, this is a more exploratory question rather than a firm test of a previously defined hypothesis. This implies that, to obtain a rather complete range of the local strategies used by different people, one needs to run more different subjects on different trials than during the testing phase of the hypothesized task structure.

Another way of obtaining relevant local strategies may be an analysis of handbooks, workprocedures and so on. Although handbooks and workprocedures almost never contain useful information regarding the task structure, they may be quite useful in terms of the identification of local strategies (for example: how to carry out a certain test).

Going back to diagnosis, for example in technical environments, several local strategies have been described. For the task of the identification of possible causes (an element of the task structure, as described before), successful local strategies described are topographic search (e.g. Rasmussen, 1986), in which an association between symptoms and underlying possible causes is made based on knowledge about the structure or functioning of the system, and the use of heuristics, by means of which, often based on experience, a direct connection is made between a symptom and its possible underlying fault. For the task of testing, another element of the task structure, split-half strategies by means of

which the number of tests to localize the fault are minimized, are rather successful.

Search strategies may be different in different situations, and may also depend on the level of expertise the person has. Since heuristics are mostly developed through practical experience and are tied to specific situations, they may become increasingly available with increasing levels of expertise. As demonstrated by Rasmussen (1986), topographic search, although not in all situations informationally economic, may be preferred by technicians in domains such as electronic troubleshooting.

Some of the search strategies that have been described in the literature may be considered more powerful strategies than others. Obviously, heuristics belong to the most powerful local strategies a person may have, although they are only applicable in a narrow domain, and have no wider generality than this domain. Therefore, they are likely to fail in any new situation. Less powerful, but still leading to conclusions rather efficiently, are search strategies such as topographic search, geared towards diagnosis in technical domains, but more widely applicable than heuristic search. At the next level of generality are search strategies such as split-half approaches. At the lowest level of specificity are problem solving methods, such as means-ends analysis and generate and test, as described by Newell and Simon (1972).

6.3 **Development of a model for the applicable domain knowledge**

Now that we have addressed how to develop a model of the task structure and the local strategies of a given task, the question remains how the relevant underlying domain knowledge may be identified. Early work on expert systems seemed to imply that there was only one underlying model of domain knowledge. Work along this line gave rise to the *model-based approach*, which postulates that expert system building should start with an encoding of the first principles of a domain, for example in diagnosis, qualitative or quantitative models of the behaviour of the device to be diagnosed (Davis, 1984; DeKleer, 1984). In most of the cases, this work concentrates on models about the structure and behaviour of the device. However, as described in Steels (1990), it is possible to think of a variety of models, each focusing on different aspects of the problem domain. For diagnosis, for example, one could think of a structural model describing part-whole relationships between components and subsystems, a causal model representing the cause-effect relationships between properties of components, a functional or behavioral model representing how the function of the whole follows from the function of the parts, a fault model representing possible faults and components for each function that might be responsible for the fault, and an associational model relating observed properties with states of the system. Which of these models would be the one to use? Clearly, all these models are useful. Simmons (1988), cited in Steels (1990), for example, describes a system that

translates causal models into associational models and shows how they have complementary utility in problem solving. Thus, the question is not what model to use exclusively, but what type of model or type of domain knowledge at what level of abstractness is appropriate in certain stages of a reasoning process.

The principal way to get a first indication of the applied domain knowledge is an analysis of protocols collected in task situations with respect to the underlying knowledge used. The pieces of knowledge collected this way are at least guaranteed to be used during problem solving, otherwise they would not have shown up in the verbal protocols. Another way of identifying relevant domain knowledge would be an investigation into training and educational material used. However, not all of this material is completely geared to the task at hand. Sometimes training materials and handbooks (partly) consist of pieces of knowledge that are supposed to be useful for trainees, but which in practice are never used anymore, and therefore should not be included in the task model. Moreover, technical documentation has not been written for educational purposes, and apart from including types of knowledge that will never be used in problem-solving situations, it is not unlikely that certain types of knowledge (for example knowledge at a higher level of abstraction) are not included in manuals or technical documentation.

Based on this type of analysis, it is possible to identify for each part of the task structure which pieces of knowledge play a role, resulting in tables such as the following (adapted from Schaafstal, 1991 (diagnosis in the domain of paper making)).

Table I Relation between phases of the task structure and used domain knowledge.
1 = Symptoms, 2 = Judgment, 3 = Possible faults, 4 = Ordering of faults, 5 = Testing, 6 = Determination of repairs, 7 = Consequences of application of repairs, 8 = Evaluation. Topogr. location = knowledge about the topographical location of a component. Function comp. = knowledge about the function of a certain component. Functioning comp. = knowledge about how the component (internally) works.

	1	2	3	4	5	6	7	8
Process flow		*	*		*		*	
Topogr. location						*		
Controls	*		*		*	*	*	
Function comp.	*	*	*		*	*		
Paper making	*	*	*	*		*	*	*
Normal values	*		*		*			*
Process dynamics	*		*		*			*
Functioning comp.			*			*		

7 USE OF THE TASK MODEL FOR DIAGNOSING AND PREDICTING HUMAN COGNITIVE BEHAVIOUR

Since the framework underlying the task model is closely related to ACT*, it should enable us to diagnose and predict human cognitive behaviour. One of the main sources for error in human behaviour stems from an overload of the working memory system. In the task model outlined above a working memory overload may show up in various locations. First, since the model is of a hierarchical nature, especially novices may have difficulties in keeping track of all the various subgoals that they have to set up in a specific situation. For example in diagnosis, when having to test a certain hypothesis, novices may not know exactly how to perform this test, having to set up a separate subgoal for finding this in the documentation. After they have found out how to perform a certain test, the second problem (next subgoal at a deeper level) may consist of finding the reference values that they should look for, and finally, at the deepest level, they may have to set a subgoal for the search of applicable domain knowledge. Therefore, it is not unlikely that, due to this extensive subgoaling which may lead to a working memory overload, novices will have difficulties keeping on track during problem solving.

A second problem that may occur, already included in the previous, is when domain knowledge is not available in a format usable during problem solving. Thus, one would predict that when the right domain knowledge is unavailable, problem solving will eventually fail. Novices may also lack the relevant global and local strategies (see Chapter 4 above), in which case their behaviour, compared to expert behaviour, may look rather unsystematic. However, unavailability of the good strategies could also lead to novices getting stuck in a very early phase of problem solving: they may not know how to get started on the task.

The model enables an analysis of expert-novice differences in terms of differences in strategy applied, procedures used, and domain knowledge used. By means of this difference it is possible to analyze misconceptions that novices may have about the domain as well as buggy procedures.

The model predicts that experts behave according to what the model describes: they will basically follow the strategy steps in the prescribed order. Since experts will have compiled parts of their knowledge it may be that the domain knowledge used is not always stated very clearly, but has to inferred from (verbal) behaviour.

8 USE OF THE TASK MODEL FOR THE DEVELOPMENT OF KNOWLEDGE-BASED SYSTEMS

An important part of the development of knowledge-based systems is the knowledge gathering and knowledge structuring process. In this phase, the knowledge available about a certain type of problem is gathered and structured, in such a way that it can be efficiently implemented afterwards. The knowledge gathered may come from several sources: handbooks, (training)manuals, documentation, and obviously, human experts, and a frequently heard problem in this respect is the problem of the integration of the various knowledge sources. Therefore, one needs a good interpretation scheme, which will serve as a guideline for the knowledge gathering and structuring process, thus enabling the interpretation and structuring of the acquired knowledge.

As stated above, knowledge structuring is dependent on a good interpretation scheme that will serve as a guideline for the structuring process. These interpretation schemes are focused at the *task level*. Knowledge at the task level concerns knowledge about the global steps taken by a person when he/she is doing the task. This is often referred to as the "global strategy" followed in performing the task, the task structure as it was defined before. Knowledge at the domain level concerns knowledge about the domain at hand needed to obtain information for carrying out the steps defined at the task level. For example, in diagnostic tasks, two steps at the task level are "identify symptoms", and "determine possible faults for this (set of) symptom(s)". At domain level, for example in the paper industry, the identification of symptoms may be accomplished by looking at the paper, by reading off alarms, by messages from other people, and so on. The determination of possible faults for a set of symptoms is accomplished by a search process through the machine (or memory), leading to a list of possibilities. Holes in the paper (a symptom), for example, may be caused by a number of underlying faults in the paper making process, such as slight damages to parts of the installation, wrong settings and so on (determination of underlying faults).

The model for the generic task of diagnosis as proposed by Schaafstal (1991) has been used as interpretation scheme for the development of various knowledge-based systems for diagnosis in technical environments (Schaafstal & Bogers, 1990; Schaafstal & Bogers, 1991). From the projects carried out it turned out that this task level description of diagnosis proved to be a fruitful framework for structuring and interpreting the knowledge we obtained, since it makes it possible to classify the different knowledge fragments experts come up with. This is especially important in the not exceptional situation that experts do not bring their knowledge to bear in a very systematic way.

However, the framework cannot only be used as a tool for knowledge gathering, it also provides support in deciding whether the information gathered may be considered complete and supports the developer in keeping track of progression made. It may also serve as the basic structuring tool in the implementation

process, resulting in systems that are easier to understand and to maintain. Therefore, we strongly recommend using a framework such as this in knowledge-based system development, since it greatly enhances the efficiency of development, and thus reduces the costs.

Another way of using a previously developed task model in the development of knowledge-based systems would be an analysis of the difficulties people of different levels of expertise have with the task. In this situation, a match is made between elements of the task model at all three levels and the gaps that may be identified in someone's knowledge and use of strategies in a certain task. By means of this analysis a more principled analysis of the needs of a user is possible, resulting in knowledge-based systems doing a better job in supporting people than some of the ones built in a less principled way.

Finally, a task model opens the possibilities to develop a knowledge-based system based on various types of knowledge that have previously been identified as playing an important role in the task. One of the issues in the development in so-called "second-generation expert systems" concerns the incorporation of models of domain knowledge into the system, thus resulting in less brittle systems able to reason not only with rather superficial rules, but also with the underlying domain knowledge. In most cases the incorporated domain knowledge concerns knowledge about the structure and behaviour of the system, and one may wonder whether it would not be beneficial to incorporate many different types of knowledge that play a role in the domain.

9 USE OF THE TASK MODEL FOR TRAINING ISSUES

An interesting issue with respect to the task model is the question whether it can be used for training of complex cognitive tasks, such as diagnosis. Before this point will be addressed, it is worth mentioning some comments about the present way of training complex cognitive tasks. An important feature of many of the current training programs is the heavy emphasis on the acquisition of domain or system knowledge. However, this is too often approached using theoretical principles or formal scientific laws that define the domain. Results of several studies, as summarized in Morris and Rouse (1985) indicate that instruction in theoretical principles is not an effective way to produce good troubleshooters. Explicit training in theories, fundamentals or principles fails to enhance and sometimes actually degrades diagnostic performance (Brigham & Laios, 1975; Crossman & Cooke, 1974; Kragt & Landeweerd, 1974; Morris & Rouse, 1985; Mayer & Greeno, 1972; Mayer, Stiehl, & Greeno, 1975). However, one cannot carry out a task without having system knowledge at all, but the types of system knowledge trained should be geared towards the job, which implies that domain knowledge should be represented in such a way that it facilitates problem solving (more levels of abstraction, better differentiation (see table 1)). Despite the

importance of efficient strategies for good task performance, not much effort is put into explicit training of good strategies at all. This may partly be due to the misconception that training in system knowledge will automatically result in good task performance, since the two are closely linked. It may also stem from the idea that good strategies will automatically evolve with increasing experience on-the-job, and therefore explicit training will not help all that much. In itself, this idea is valid: increasing experience with a certain task most of the times leads to a better performance. However, the question is whether training as a whole can be speeded up and be made more efficient if strategy training is taken into account. A final reason for the absence of strategy training is the fact that good strategy training appears difficult to accomplish, which makes it hard to incorporate strategy training in regular operator training courses.

Having discussed some of the issues in many current training programs for complex cognitive skills, we will continue with a discussion about the topic whether the task structure can be used for strategy training. The task structure basically consists of a goal structure, in which every step or subtask as defined can be regarded a goal. Thus, the model offers a guideline and opportunities for training explicit goal structures. This is especially interesting since, although importance of strategic control knowledge in the form of goal structures has been shown in a number of domains (Greeno, 1978; Card, Moran, & Newell, 1983; Anderson, Farrell, & Sauers, 1984; Kieras, 1987; Gott & Pokorny, 1987), it happens that in instructional materials a number of times gaps have been identified with respect to goal structures (Greeno, 1978; Anderson et al., 1984; Kieras, 1987). The emphasis on the importance of training people how to use strategic knowledge is also one of the most important outcomes of the Guidon project (Clancey, 1987). Training of explicit goal structures is not impossible, though, since it has already been accomplished a number of times, for example in some of the intelligent tutoring systems, such as Proust, a knowledge-based cognitive diagnosis program that identifies programming bugs with an emphasis on bugs regarding the abstract program plan (Soloway & Johnson, 1984), Bridge, a tutorial learning environment that concentrates on the planning knowledge required early in programming skill acquisition (Bonar, 1985), and the Lisp tutor (Anderson & Reiser, 1985). For Proust, it has been shown that its effectiveness in diagnosing programming errors on simple Pascal problems is comparable to that of human teaching assistants, that is, about 75% correct. The Lisp tutor has been evaluated involving comparisons of various instructional alternatives to learning Lisp, including traditional classroom instruction, a private human tutor and independent self-study. In these studies, the Lisp tutor is not far behind human tutors in effectiveness. The self-study alternative was much worse, particularly as the instructional material became more difficult. When a group of students used the Lisp tutor for problem exercises to supplement standard lectures, they spent 30% less time on the problems than students working on their own, but scored 43% better on the post-test (Anderson, Boyle, & Reiser, 1985). Pirolli and Anderson (1985) examined the particular effectiveness of teaching a Lisp goal structure as a strategy for learning recursive functions as

compared to "process" instructions that explained how recursive functions work, but did not teach specific steps in writing them. The "structure" group was 32% faster than the "process" group in correctly generating an initial set of functions. Anderson (1987) argues that instruction in a skill is most effective when it directly provides information needed in a production-system model of that skill. This information includes the goal structure as well as procedural steps. The results regarding these computer tutors may be considered promising with respect to strategy training.

With respect to training troubleshooting procedures, the second level of the task model, quite some research has been carried out on the effect of *proceduralized training*. Proceduralized training represents an extreme form of rule-based troubleshooting instruction. The student is provided with exact step-by-step procedures at a very concrete level to learn and practice during training sessions. The theory of performance that underlies this approach is problem solving as "the rote execution of procedural steps". Learning is assumed to occur via rote practice and general physical familiarity with the system. Potter and Thomas (1976), in a study regarding troubleshooting performance in a technical domain, showed that fully proceduralized job aids resulted in superior performance compared to technical orders or logical troubleshooting aids. This effect was strongest for simpler fault isolation tasks and for less experienced technicians. This suggests the same kind of instructional applicability as seen in the Lisp tutor (Pirolli & Anderson, 1985) where a complete prespecification of steps appears feasible and effective on easier tasks in the initial stages of learning, but may loose effectiveness for more difficult tasks. This result may be explained by the absence of an explicit goal structure in proceduralized training, while an explicit hierarchically organized goal structure is one of the characteristics of highly efficient expert behaviour, which will especially be called upon in more complex situations. A pragmatic limitation to the proceduralization approach is the fact that one can seldom anticipate all events or combinations of events that may occur in a particular system. Thus, operators will inevitably encounter events for which there is no such procedure, or events in which it is unclear which procedure, if any, should be used. In such situations, proceduralized training is of no use (Rouse, 1982). Morris and Rouse (1985) therefore conclude their review of research in troubleshooting with the recommendation that the most promising instructional approach worthy of further research is one where procedures for approaching fault isolation problems are combined with instruction in how to use system knowledge in deciding exactly what to do.

Translating this to the training of complex cognitive skills leaves us with the following recommendations. First, domain knowledge to be trained should be geared towards job contents, and thus, a thorough analysis of job contents is of crucial importance. This implies that, for example, the domain knowledge taught to technicians and operators may be rather different in nature. For operators, it is especially important to know aspects such as the function that certain process components have on the paper making process, but they should not be bothered

with too many technical details. Technicians on the other hand, need a thorough understanding of components at a technical level, and their knowledge about the influence of components on paper making may be more superficial. Second, much more effort should be put into training of explicit task-level strategies, since such an efficient goal structure appears to be almost completely lacking in novice behaviour (Schaafstal, 1991).

REFERENCES

Anderson, J.R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Anderson, J.R. (1987). Skill acquisition: compilation of weak-method problem solutions. Psychological Review, 94, 192-210.

Anderson, J.R., Boyle, C.F., Farrell, R.G. & Reiser, B.J. (1984). Cognitive principles in the design of computer tutors. In *Proceedings of the Sixth Cognitive Science Society Conference.* Boulder, CO.

Anderson, J.R. & Reiser, B.J. (1985). The LISP tutor. *Byte, 3,* 159-175.

Anzai, Y. & Simon, H.A. (1979). The theory of learning by doing. *Psychological Review, 86,* 124-140.

Bainbridge, L. (1989). Cognitive task analysis for process operations: A model of cognitive processes, and its implications. Second discussion draft, October 1989, Ergonomics Workgroup, University of Twente, The Netherlands.

Bonar, J.G. (1985). *Mental models of programming loops.* Technical report, Pittsburgh, PA: University of Pittsburgh, Learning Research and Development Center.

Bovair, S., Kieras, D.E. & Polson, P.G. (1990). The acquisition and performance of text-editing skill: A cognitive complexity analysis. Human-Computer Interaction, 5, 1-48.

Brown, D.C. & Chandrasekaran, B. (1986). Knowledge and control for a mechanical design expert system. IEEE Computer, 19(7), 92-100.

Card, S.K., Moran, T.P. & Newell, A. (1983). The psychology of human-computer interaction. Hillsdale, NJ: Lawrence Erlbaum.

Chandrasekaran, B. (1983). Towards a taxonomy of problem solving types. AI Magazine, 4(1), 9-17.

Clancey, W.J. (1987). *Knowledge-based tutoring. The GUIDON program.* Cambridge, MA: MIT Press.

Drury, C.G., Paramore, B., Van Cott, H.P., Grey, S.M. & Corlett, E.N. (1987). Task analysis. In G. Salvendy (Ed.), Handbook of human factors (pp. 371401). New York: Wiley.

Ericsson, K.A. & Simon, H.A. (1984). Protocol analysis: Verbal reports as data. Cambridge, Mass.: MIT Press.

Ernst, G.W. & Newell, A. (1969). *GPS: A case study in generality and problem solving.* New York: Academic Press.

Essens, P.J.M.D. & McCann, C.A. (1991). Human cognitive processes in command and control planning. 3: Determining basic processes involved in planning in time and space. Soesterberg: TNO Institute for Perception, Report IZF 1991 B-11.

Friedland, P.E. & Iwasaki, Y. (1985). The concept and implementation of skeletal plans. Journal of Automated Reasoning, 1, 161-208.

Goel, V. & Pirolli, P. (1989). Motivating the notion of generic design within information processing theory: the design problem space. AI Magazine (Spring), 18-36.

Gott, S.P. (1989). Apprenticeship instruction for real-world tasks: The coordination of procedures, mental models, and strategies. In E.Z. Rothkopf (ed.), *Review of Research in education, 15*. Washington, DC: AM. Educ. Res. Assoc.

Gott, S.P. & Pokorny, R. (1987). The training of experts for high-tech work environments. In *Proceedings of the Ninth Interservice/Industry Training Systems Conference.* Washington, DC.

Greeno, J.G. (1978). A study of problem solving. In R. Glaser (Ed.), *Advances in instructional psychology: Vol. 1.* Hillsdale, NJ: Lawrence Erlbaum.

Grant, S. & Mayes, T. (1991). Cognitive task analysis? In G.R.S. Weir & J.L. Alty (Eds.), Human-Computer interaction and complex systems. London: Academic Press.

Hamel, R. (1990). Over het denken van de architect. (in Dutch, with an English sum-mary). PhD-Thesis, University of Amsterdam.

Holyoak, K.J. (1990). Symbolic connectionism: Toward third-generation theories of expertise. Technical Report UCLA-CSRP-90-14, University of California, Los Angeles.

Kieras, D.E. & Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cognitive Science, 8*, 255-273.

Kieras, D. (1991). A guide to GOMS task analysis. Prepared for EECS 493 - User Interface Design and Analysis and CHI'91 Tutorial: How to Do a GOMS Analysis for Interface and Documentation Design.

Laird, J.E., Newell, A. & Rosenbloom, P.S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

Larkin, J.H., Reif, F., Carbonell, J. & Gugliotta, A. (1988). FERMI: A flexible expert reasoner with multi-domain inferencing. *Cognitive Science, 12*, 101-138.

Malhotra, A., Thomas, J.C., Carroll, J.M. & Miller, L.A. (1980). Cognitive processes in design. International Journal of Man-Machine Studies, 12, 119-140.

Marcus, S., Stout, J. & McDermott, J. (1988). VT: An expert elevator designer that uses knowledge-based backtracking. AI Magazine, 9, 95-112.

Mittal, S., Dym, C.L. & Morjaria, M. (1986). PRIDE: An expert system for the design of paper handling systems. IEEE Computer, 19(7), 102-114.

Olson, J.R. & Olson, G.M. (1990). The growth of cognitive modelling in human-computer interaction since GOMS. Human-Computer Interaction, 5, 221-265.

Pirolli, P.L. & Anderson, J.R. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology, 39*, 240-272.

Rasmussen, J. (1986). *Information processing and human-machine interaction. An approach to cognitive engineering.* Amsterdam: Elsevier Science Publishers.

Schaafstal, A.M. (1991). Diagnostic skill in process operation: A comparison between experts and novices. PhD-Thesis, University of Groningen, The Netherlands.

Schraagen, J.M.C. (1990). How experts solve a novel problem within their domain of expertise. Soesterberg: TNO Institute for Perception, Report IZF 1990 B-14.

Schaafstal, A.M. & Bogers, H.H. (1990). Kennissysteem behulpzaam bij diagnose van technische storingen. *PT-Procestechniek, 45 (april 1990)*, 48-51.

Schaafstal, A.M., & Bogers, H.H. (1991). The development of knowledge-based systems for paper machine diagnostics. Problem or Pleasure? *Proceedings of the Pira International Conference on Paper machine diagnostics.*

Schoenfeld, A.H. (1979). Explicit heuristic training as a variable in problem-solving performance. *Journal of Research in Mathematics Education, 10*, 173-187.

Shepherd, A. (1985). Hierarchical task analysis and training decisions. *Programmed learning and educational technology, 22*, 162-176.

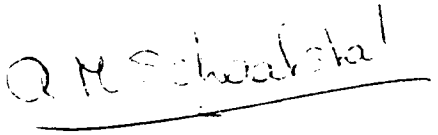
Simmons, R. (1988). *Generate, test and debug: a paradigm for solving interpretation and planning problems*. PhD-thesis, AI lab, Massachusetts Institute of Technology.

Soloway, E.M., & Johnson, W.L. (1984). Remembrance of blunders past: A retrospective on the development of PROUST. *Proceedings of the Sixth Cognitive Science Society Conference.*

Steels, L. (1990). Components of expertise. AI Magazine, 11(2), 28-49.

Wognum, P.M. (1990). Explanation of automated reasoning: How and why? PhD- Thesis, University of Twente, The Netherlands.

Soesterberg, July 24, 1992



Dr. A.M. Schaafstal

APPENDIX 1 Practical guidelines for carrying out protocol analysis

A protocol analysis is particularly useful for identifying the global and local strategies that are important in carrying out particular tasks. Identification of relevant domain knowledge is also possible. However, if one wants to study the structure of the domain knowledge in more detail, other techniques than protocol analysis should be used.

When carrying out a protocol analysis, it is important to remember that the analyst should interpret the verbal data obtained, not the subject who performed the task while thinking aloud. The subject's verbalizations should be considered as data, just as reaction times or percentage correct. And just as reaction times need a theoretical framework in order to be interpreted, so verbal data need a framework in order to be interpretable. It is the analyst's task to provide that framework. Bearing this in mind, the following steps may be distinguished in order to arrive at a framework in which verbal data can be interpreted:

1 Normative or rational task analysis
2 Descriptive psychological model
3 Coding scheme

The first step is called a normative or rational task analysis because it specifies the task goals that are required for optimal, normative, task performance. These goals may be derived from various sources: intuition, handbooks, interviews with expert practitioners, or by examining how similar types of tasks are carried out (in other words, by studying generic tasks). The results of this step are very similar to what we have discussed above under the heading of AI-approaches to generic tasks.

The second step specifies the human capacities and limitations pertaining to the task. By taking these capacities and limitations into account, the output of the rational task analysis is modified. When studying experts, these modifications are often slight, since experts are not bounded as much by limitations as novices are. As a first approximation, it is often sufficient to use the normative task goals and put a particular ordering on them. In this way, a model is specified that posits a number of activities that are carried out in a particular order.

The third step is to convert the psychological model into a coding scheme. This is necessary in order to be able to classify actual protocol statements. For instance, subjects will often comment on the task's difficulty or on their own progress or thought processes. These comments are outside the scope of the psychological model, but should be incorporated in the coding scheme. In other words, the coding scheme adds non-taskspecific statements to the psychological model. The aim should be a coding scheme that is sufficiently rich so as to capture all possible verbal statements. However, the coding scheme should not be so detailed that statements are difficult to assign to the various categories. The optimal number of categories that a coding scheme should contain is

difficult to determine. This depends on one's purposes and on the particular task involved. However, based on our own experience, somewhere between 3 and 15 categories seems appropriate.

It is appropriate at this stage to collect a number of protocols in a pilot study, and try to assign protocol statements to the various categories. Give a few examples from a protocol for each category. This aids in the coding of later protocols.

It is sometimes possible by using 'key words' appearing in protocol statements to assign statements to categories by means of these key words. For instance, if a subject says: "I will now try to add these numbers", the use of the word "will" indicates the setting of a goal, namely to add numbers. Using key words adds to the objectivity and reliability of scoring.

Once one has developed a coding scheme, the psychological model can be tested against empirical data. Questions that may be answered by the data are the following:

1 Are cognitive activities missing in the model?
2 Are cognitive activities superfluous in the model?
3 Do the data support the order of the activities in the model?

The actual process of testing the model against the data involves the following steps:

1 Select empirical data
1.1 Justify use of verbal protocols
1.2 Instruct subjects appropriately
1.3 Collect verbal protocols
2 Carry out exploratory protocol analysis
2.1 Randomly select a small number of protocols
2.2 Transcribe protocols
2.3 Segment protocols
2.4 Assign protocol segments to categories
2.5 Revise model, if necessary
3 Test model
3.1 Transcribe remaining protocols
3.2 Segment protocols
3.3 Randomize protocol segments
3.4 Select naive coders (at least 2)
3.5 Assign protocol segments to categories
3.6 Determine inter-rater reliability

Below, the steps will be discussed in more detail.

1 Select empirical data

1.1 Justify use of verbal protocols

Verbal protocols are not appropriate when:

a information in working memory is difficult to verbalize (e.g., when the task is highly visual)
b the task is highly practiced
c subjects have a low verbal intelligence
d subjects have to work under time pressure

1.2 Instruct subjects appropriately

a the appropriate kernel instruction is: "Please think aloud while you are carrying out your task"; do NOT give the following instruction: "Please tell me what you think while you are carrying out your task". This invites theorizing on the part of the subject and reflection on their own thought processes. Remember this is part of the analyst's task, not the subject's.
b let subjects practice with thinking aloud; give them a few practice problems as suggested by Ericsson and Simon (1984).

1.3 Collect verbal protocols

a use cassette recorder to record subject's verbalizations; obtain subject's permission to record their verbalizations.
b make sure the cassette recorder works.
c give subjects a written problem; have them read the problem aloud.
d make sure subjects understand both the problem and their task of thinking aloud.
e when subjects fall silent for more than 10 s during problem solving, prompt them by saying: "Please think aloud". Do NOT prompt them by saying: "Please tell me what you are thinking now".
f code each cassette so as to be able to retrieve the appropriate subject and the appropriate problem; i.e., write down subject's name (or a code for the name if anonymity is required), the problem number and counter indication (e.g., problem 1: 000-678; problem 2: 679-973), the date and time of day, the experimenter's name (if necessary), the experimental condition (if appropriate).

2 Carry out exploratory protocol analysis

a the goal of the exploratory protocol analysis is to check the sufficiency of the coding scheme
b this goal should always be accomplished by using a separate set of protocols, either from a pilot study or from a small number of protocols from the final study; if protocols from the final study are used, they cannot be used again.

2.1 Randomly select a small number of protocols

2.2 Transcribe protocols

a protocols should be transcribed literally from the audiotapes; pauses are just as important as speech; indicate pauses by dots (e.g., one dot for every 5 s of pause).
b if video and audio are separately used, make sure both traces are synchronized in one running protocol.

2.3 Segment protocols

a a frequently used method for segmenting protocols is the one based on pauses in speech, i.e., after each pause, a new segment starts.
b number each protocol segment.

2.4 Assign protocol segments to categories

a assign each numbered protocol segment to a category from the coding scheme; if possible, use the example protocol statements and the key words given with each category.

2.5 Revise model, if necessary

a check what protocol segments cannot be assigned to any category; add categories, if necessary
b check what categories are superfluous; delete categories, if necessary
c construct a mathematical model that predicts the number of transitions among categories, and test the model via parameter estimation techniques against the number of observed transitions; if the predicted and observed frequencies significantly differ, as indicated by a Chi-square value, revise the model.

3 Test model

3.1 Transcribe remaining protocols

- as described in 2.2 above

3.2 Segment protocols

- as described in 2.3 above

3.3 Randomize protocol segments

a randomization is required in order to assign each segment independent of another segment to a category; in practice, however, the meaning of statements is often only apparent from the context in which they appear, so that randomization would be counterproductive; therefore, the analysts should use their own judgment whether to randomize or not;
b randomize by using a series of random numbers and match the protocol numbers to the random numbers

3.4 Select naive coders (at least 2)

a 'naive' coders are coders who are unaware of the hypotheses that are tested, but are sufficiently familiar with the domain under consideration;
b train coders on the protocols selected for exploratory protocol analysis

3.5 Assign protocol segments to categories

- as described in 2.4 above

3.6 Determine inter-rater reliability

a assess, for each category, the number of times raters agree on assignment of segments to categories, and the number of times they disagree.
b calculate the percentage agreement between raters (should be higher than 80%), or calculate Cohen's Kappa (should be higher than .70).

REPORT DOCUMENTATION PAGE

1. DEFENCE REPORT NUMBER (MOD-NL)	2. RECIPIENT'S ACCESSION NUMBER	3. PERFORMING ORGANIZATION REPORT NUMBER
TD 92-2278		IZF 1992 B-5
4. PROJECT/TASK/WORK UNIT NO.	**5. CONTRACT NUMBER**	**6. REPORT DATE**
788.3	B92-34	July 24, 1992
7. NUMBER OF PAGES	**8. NUMBER OF REFERENCES**	**9. TYPE OF REPORT AND DATES COVERED**
38	44	Final

10. TITLE AND SUBTITLE

A method for cognitive task analysis

11. AUTHOR(S)

A.M. Schaafstal and J.M.C. Schraagen

12. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

TNO Institute for Perception
Kampweg 5
3769 DE SOESTERBERG

13. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

TNO Defence Research
Schoemakerstraat 97
2628 VK Delft

14. SUPPLEMENTARY NOTES

15. ABSTRACT (MAXIMUM 200 WORDS, 1044 BYTE)

A method for cognitive task analysis is described based on the notion of "generic tasks". The method distinguishes three layers of analysis. At the first layer, the task structure, top-level goals of a certain task are identified that have to be fulfilled during task-execution. This task structure may also be viewed as the global strategy to carry out the task. At the second layer of analysis, the local strategies (procedures) are identified by means of which values are obtained for goals in the task structure. The third layer of analysis consists of a description of the underlying domain knowledge. After a general discussion of the potentialities of the task model in predicting and diagnosing human cognitive behaviour, implications of the model for applied areas such as the development of knowledge-based systems and training, are discussed.

16. DESCRIPTORS	IDENTIFIERS
Problem solving	Cognitive Task Analysis Knowledge Strategies

17a. SECURITY CLASSIFICATION (OF REPORT)	17b. SECURITY CLASSIFICATION (OF PAGE)	17c. SECURITY CLASSIFICATION (OF ABSTRACT)
18. DISTRIBUTION/AVAILABILITY STATEMENT		17d. SECURITY CLASSIFICATION (OF TITLES)
Unlimited availability		